# Approximating Material Interfaces during Data Simplification

David E. Sigeti* Benjamin F. Gregorski†
John Ambrosiano * Gerald Graham * Murray Wolinsky *
Mark A. Duchaineau‡ Bernd Hamann † Kenneth I. Joy †

† Center for Image Processing and Integrated Computing (CIPIC), Department of Computer Science, University of California, Davis, CA 95616-8562, USA

‡ Center for Applied Scientific Computing (CASC) Lawrence Livermore National Laboratory, P.O. Box 808, L-561, Livermore, CA 94551, USA
Lawrence Livermore National Laboratory

* Los Alamos National Laboratory
Los Alamos, New Mexico 87545

## Abstract

We present a new method for simplifying large data sets that contain material interfaces. Material interfaces embedded in the meshes of computational data sets are often a source of error for simplification algorithms because they represent discontinuities in the scalar or vector field over a cell. By representing material interfaces explicitly in a data simplification process, we are able to provide separate field representations for each material over a single cell. Our algorithm uses a multiresolution tetrahedral mesh supporting fast coarsening and refinement capabilities; error bounds for feature preservation; explicit representation of discontinuities within cells; and separate field representations for each material within a cell.

## 1 Introduction

Computational physics simulations are generating larger and larger amounts of data. They operate on a wide variety of input meshes, including rectilinear meshes, adaptively refined meshes for Eulerian hydrodynamics, unstructured meshes for Lagrangian hydrodynamics and arbitrary Lagrange-Eulerian meshes. Very often, these data sets contain special physical features such as material interfaces, physical boundaries, or thin slices of material that must be preserved when the field is simplified. In order to ensure that these features are preserved, the simplified version of the data set needs to be constructed using strict $L^\infty$ error bounds that prevent small yet important features from being eliminated.

Data sets of this type require a simplification algorithm that is capable of approximating data sets with respect to several simplification criteria. The cells in the approximation must satisfy error bounds with respect to the dependent field variables over each mesh cell, and to the representation of the discontinuities within each cell. In addition, the simplification algorithm must be able to deal with the wide range of possible input meshes as described above.

*{ambro,ggraham,murray,sigeti}@lanl.gov
†{gregorsk,hamann,joy}@cs.ucdavis.edu
‡{duchaineau1}@llnl.gov

We present an algorithm for generating an approximation of a computational data set that can be used in place of the original high-resolution data set generated by the simulation. Our approximation is a resampling of the original data set that preserves user-specified as well as characteristic features in the data set and approximates the dependent field values to within a specified tolerance.

The basis for our simplification algorithm is the subdivision of a tetrahedral mesh as presented by Zhou et al. [1]. We generalize their implementation by removing the restriction that the input data needs to be given on a regular rectilinear mesh consisting of $(2^N+1)$x$(2^N+1)$x$(2^N+1)$ cells. Given a data set and polygonal representations for the material interfaces, our algorithm constructs an approximation as follows:

1. Our algorithm starts with the base mesh of 6 tetrahedra and associates with each one the interface polygons that intersect it.
2. The initial tetrahedral mesh is first subdivided so that the polygonal surface meshes describing the material interfaces are approximated within a certain tolerance. At each subdivision, the material interface polygons lying partially or entirely in a cell are associated with the cell's children; approximations for the polygons in each child cell are constructed, and interface approximation errors are computed for the two new child cells.
3. The mesh is further refined to approximate the field of interest, e.g., density or pressure within a specified tolerance.

For the cells containing material interfaces, our algorithm computes a field representation for each material. This is done by extrapolating *ghost* field values for each material at each vertex. When the field approximation error for the cell is computed, the separate field representations, built using these ghost field values, are used to calculate an error for each distinct material in the cell. The decomposition process of a cell that contains multiple materials consists of these steps:

1. The signed distance values and ghost values for the new vertex are computed when the vertex is created during

a split or subdivision operation. This is done by examining those cells that share the edge being split.

2. The interface representations, i.e., triangle meshes, are associated with the child cells, and the approximating interfaces representations and their associated errors are computed.

3. The field error for each of the materials is computed, and the maximum value of these errors is the overall error associated with a cell containing multiple materials.

## 2 Related Work

Hierarchical approximation techniques for triangle meshes, scattered data, and tetrahedral meshes have matured substantially over recent years. In [2], Hoppe describes the progressive mesh simplification method for triangle meshes. An arbitrary mesh is simplified through a series of edge collapse operations to yield a very simply base mesh. The ROAM system, described in [3], uses priority queue-driven split and merge operations to provide optimal real-time display of triangle meshes for terrain rendering applications. The tetrahedral mesh structure used in our framework is an extension of the original ROAM data structure for triangle meshes. Heckel and Uva [4], [5] describe methods for constructing surface hierarchies based on adaptive clustering. Multiresolution methods for reconstruction and simplification have also been explored using subdivision techniques and, especially, wavelets.

Simplification of tetrahedral meshes has been explored in [1], [6], [7], [8], and [9]. Zhou et al. [1] present the multiresolution tetrahedral framework that is the basis of our simplification algorithm. (This is further discussed in Section 2). Cignoni at al. [6] describe a multiresolution tetrahedral mesh simplification technique built on scattered vertices obtained from the initial dataset. Their algorithm supports interactive level-of-detail selection for rendering purposes. Trotts et al. in [7] simplify tetrahedral meshes through edge collapse operations. They start with an initial high-resolution mesh that defines a linear spline function and simplify until a specified tolerance. Staadt and Gross [9] describe Progressive Tetrahedralizations as an extension of Hoppe's [2]. They simplify a tetrahedral mesh through a sequence of edge collapse operations. They also describe error measurements and cost functions for preserving consistency with respect to volume and gradient calculations and techniques for ensuring that the simplification process does not introduce artifacts such as intersecting tetrahedra.

Different error metrics for measuring the accuracy of simplified tetrahedral meshes have been proposed. Lein et al. [10] present a simplification algorithm for triangle meshes using the Hausdorff distance as an error measurement. They develop a symmetric adaption of the Hausdorff distance that is an intuitive measure for surface accuracy. In [11], Heckbert and Garland use a quadric error metric for surface simplification. They use vertex pair collapse operations to simplify triangle meshes and they use quadric matrices that define a quadric object at each vertex to control the error of the simplified surfaces.

## 3 Multiresolution Framework

A multiresolution framework for simplifying numerically simulated data needs to be a fairly robust and extensible system. It must be capable of supporting the wide range of possible input structures used in the simulations and the wide range of output data generate by these simulations. The following properties and operations are desirable for such a framework:

1. **Interactive transition between levels of detail**. The ability to quickly move between different levels of detail allows the user to select a desired approximation at which to perform a calculation (for a visualization application). Since the level of detail is directly related to the approximation error, the user can balance computation time and accuracy.

2. **Strict $L^\infty$ error bounds**. Strict error bounds prevent small yet important features from being averaged or smoothed out by the simplification process.

3. **Local and adaptive mesh refinement and local error computations**. Local mesh refinement and computations allow the representation to be refined only in the areas of interest while keeping areas of little interest at relatively lower resolutions. This is essential for maintaining interactivity and strict cell count on computers with limited resources. Local error computations are needed for efficiency. Considering data sets consisting of millions or billions of cells, the error calculations should not involve a large amount of original data.

4. **Accommodating different meshes**. Computational simulations are done on a large variety of mesh structures, and it is cumbersome to write a multiresolution algorithm for each specific structure. In order for a framework to be useful it should be easily adaptable to a broad class of input meshes.

5. **Explicit representation of field and/or material discontinuities**. Discontinuities are very important in scientific data sets and very often need to be preserved within a certain tolerance during data simplification. A multiresolution framework should support the explicit representation and approximation of these discontinuities for the different resolution levels.

Our multiresolution recursive tetrahedral framework satisfies these design criteria. Tetrahedral cells are the simplest of the polyhedral cells. This allows us to use linear basis functions to approximate the material interfaces and the dependent field variables in a cell. Since our data structure is defined recursively as a binary tree, a representation of the original data can be computed in a preprocessing step, and we can utilize methods developed for the ROAM system [3] to efficiently select a representation that satisfies an error bound or a desired cell count. This makes the framework ideal for interactive display. Strict $L^\infty$ error bounds are incorporated into the subdivision process, see [3]. The framework supports various input meshes by resampling them at the vertices of the multiresolution tetrahedral grid. Discontinuities are supported at the cell level allowing local refinement of the representations of surfaces of discontinuity in geometrically complex areas.

The framework has several advantages over other multiresolution spatial data structures such as an octree. The way we use the binary subdivision method ensures that the tetrahedral mesh will always be a conformant, i.e, all edges in the mesh end at the endpoints of other edges and not

in the interior of edges. This ensures that cracks and T-intersections will not occur anywhere in the mesh. A conformant mesh makes it easy to guarantee that representations of fields and surfaces of discontinuity are continuous across cell boundaries.

Our resampling algorithms and error bounding algorithms require that an original data set allow the extraction of the following information:

1. The values of the field variables at any point.

2. The maximum difference between our representation for a given field over one of our cells and the representation of the same field over the same volume in the original dataset

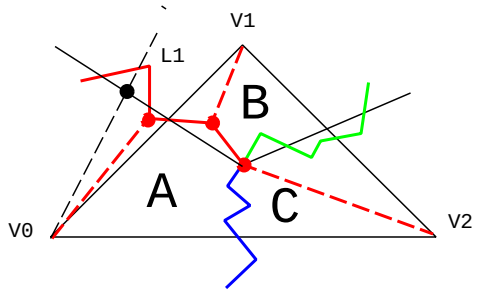


Figure 1: Triangle with three materials (A, B, and C) and three interfaces.

# 4 Material Interfaces

## 4.1 Motivation

A material interface defines the boundary between two distinct materials. Field representations across a material interface are often discontinuous. Thus, an interface can introduce a large amount of error to cells that cross it. Instead of refining an approximation substantially in the neighborhood of an interface, the discontinuity in the field is better represented by explicitly representing the surface of discontinuity in each cell. Once the discontinuity is represented, two separate functions are used to describe the dependent field variables on either side of the discontinuity. By representing the surface of discontinuity exactly, our simplification algorithm does not need to refine regions in the spatial domain with a large number of tetrahedra.

## 4.2 Extraction and Approximation

In the class of input datasets with which we are working, material interfaces are represented as triangle meshes. In the case that these triangle meshes are not known, they are extracted from volume fraction data by a material interface reconstruction technique described in [12] and [13] (The volume fractions resulting from numerical simulations indicate what percentages of which materials are present in each cell.). Such an interface reconstruction technique produces a set of crack-free triangle meshes and normal vector information that can be used to determine on which side and in which material a point in space lies.

Within one of our tetrahedra, an approximate material interface is represented as the zero set of a signed distance function. Each vertex of a tetrahedron is assigned a signed distance value for each of the material interfaces in the tetrahedron. The signed distance from a vertex $\mathbf{V}$ to an interface mesh $\mathbf{I}$ is determined by first finding a triangle mesh vertex $\mathbf{V}_i$ in the triangle mesh describing $\mathbf{I}$ that has minimal distance to $\mathbf{V}$. The sign of the distance is determined by considering the normal vector $\mathbf{N}_i$ at $\mathbf{V}_i$. If $\mathbf{N}_i$ points towards $\mathbf{V}$, then $\mathbf{V}$ is considered to be on the "positive side" of the interface, otherwise it is considered to be on the "negative side" of the interface.

Figure 1 shows a two-dimensional example of a triangle with several material interfaces and their approximations. In this figure, the red, green, and blue jagged lines are the original boundaries and the straight black lines are the approximations derived from using the signed distance values. The red dashed lines and dots show the points on the interface between materials A and B used to compute the signed distance values for the vertices $V_0$, $V_1$, and $V_2$. The black dashed line demonstrates that the projection of a point onto an approximation does not always lie inside the cell. The signed distance function is assumed to vary linearly in the cell, i.e., a tetrahedron. The distance function is a linear function $f(x, y, z) = Ax + By + Cz + D$. The coefficients for the linear function defining a boundary representation are found by solving a 4x4 system of equations, considering the requirement that the signed distance function over the tetrahedron must interpolate the signed distance values at the four vertices. Figure 2 shows a tetrahedron, a material interface approximation, and the signed distance values $d_i$ for each vertex $V_i$. The approximation is shown in cadet blue. The normal vector shown in green indicates the positive side of the material boundary approximation. Thus, the distance to $V_3$ is positive and the distances for $V_0$, $V_1$, and $V_2$ are negative.

We note that a vertex has at most one signed distance value for each interface. This ensures that the interface representation is continuous across cell boundaries. If a cell does not contain a particular interface, the signed distance value for that interface is meaningless for that cell. Given a point $\mathbf{P}$ in an interface polygon and its associated approximation $B_r$, the error associated with $\mathbf{P}$ is the absolute value of the distance between $\mathbf{P}$ and $B_r$. The material interface approximation error associated with a cell is the maximum of these distances, considering all the interfaces within the cell.

# 5 Discontinuous Field Representations

## 5.1 Motivation

Cells that contain material interfaces typically have discontinuities in the fields defined over them. For example, the density field over a cell that contains both steel and nickel is discontinuous exactly where the two materials meet. In these situations, it is better to represent the density field over the cell as two separate fields, one field for the region containing only the first material and one for the second material.

One way to accomplish this is to divide the cell into two distinct cells at the material interface. In Figure 3, the triangle would be divided into a quadrilateral for material A and a triangle for material B. The disadvantages of this method are that it introduces new cell types into the mesh and that it makes it harder to have continuous field representations across cells. Our algorithm represents the discontinuity by constructing a field representation for each material in the

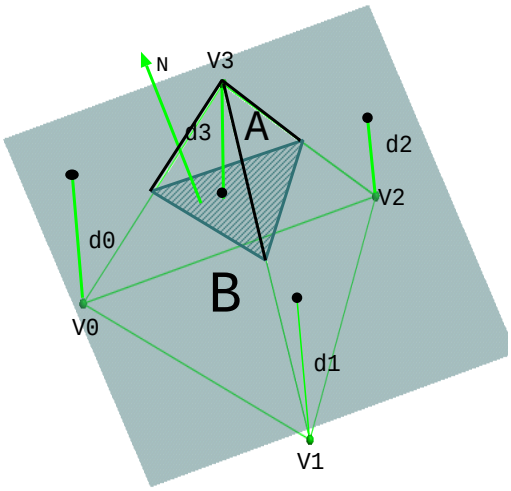


Figure 2: Tetrahedron showing signed distance values and the corresponding boundary approximation.

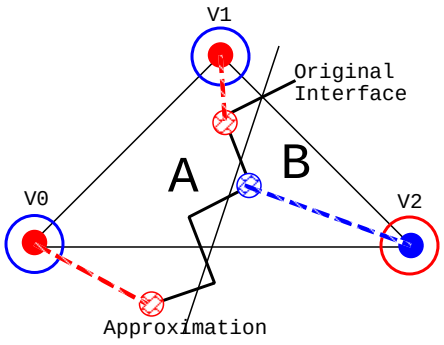


Figure 3: Triangle containing two materials.

cell. Each of the vertices in a cell must have distinct field values for each material in the cell. These extrapolated values are called ghost values.

For a vertex $\mathbf{V}$ that does not reside in material $\mathbf{M}$, we compute a ghost value for the field associated with material $\mathbf{M}$ at vertex $\mathbf{V}$. This ghost value is an extrapolation of the field value for $\mathbf{M}$ at $\mathbf{V}$. The process is illustrated in Figure 3. The known field values are indicated by the solid circles. The red circles are the known field values for material A, and the blue circle is the known field value for material B. The empty circles surrounding the filled circles indicate that a ghost value is needed for that material at the indicated vertex. Vertices $V_0$ and $V_1$ are in material A, and thus ghost values for material B must be calculated at their positions. This is indicated by the empty blue circles. Vertex $V_2$ lies in material B, and thus a ghost value for material A must be calculated at its position. This is indicated by the empty red circle.

As described in Section 1, the ghost value computation is performed when the vertex is created during the tetrahedral refinement process.

### 5.2 Computation of Ghost Values

The ghost values for a vertex $\mathbf{V}$ are computed as follows:

1. For each material interface present in the cells that share the vertex, find a vertex $V_{min}$ in a triangle mesh representing an interface with minimal distance to $\mathbf{V}$. In Figure 3, these vertices are indicated by the dashed lines from $V_0$, $V_1$, and $V_2$ to the indicated points on the interface.

2. Evaluate the data set on the far side of the interface at $V_{min}$ and use this as the ghost value at $\mathbf{V}$ for the material on the opposite side of the interface.

Only one ghost value exists per material and vertex. This ensures that the field representations are $C^0$-continuous across cell boundaries. For example consider vertex $V_0$ of the triangle in Figure 1. The vertex $V_0$ lies in material A, and therefore we must compute ghost values for materials B and C at vertex $V_0$. The algorithm will examine the three material boundaries and determine the points from materials B and C that are closest to $V_0$. The fields for materials B and C are evaluated at these points, and these values are used as the ghost values for $V_0$.

## 6 Error Metrics

The error metrics employed in our framework are similar to the nested error bounds used in the ROAM system. Each cell has two associated kinds of error values, field errors and material interface errors. In order to calculate the field errors for a leaf cell in our tetrahedral mesh hierarchy, we assume that the original data set can be divided into *native data elements*. Each of these is presumed to have a well-defined spatial extent and a well-defined representation for each field of interest over its spatial domain. The simplest example of a native data element is just a grid point that holds field values. Other possibilities are blocks of grid points treated as a unit, cells with a nonzero volume and a field representation defined over the entire cell, or blocks of such cells. Each leaf cell in our multiresolution mesh maintains links to the native data elements with which it intersects. We assume that it is possible to bound the difference between our representation of a given field over one of our leaf cells and the representation of the same field over each of the native data elements with which the given cell intersects. The error for the given field in the given cell is then just the maximum of the errors associated with each of the intersecting native data elements. Currently, we are dealing only with native data elements that are grid points of zero volume.

The field error $e_T$ for a non-leaf cell is computed from the errors associated with its two children according to:

$$e_T = \max\{e_{T_0}, e_{T_1}\} + |z(v_c) - z_T(v_c)| \qquad (1)$$

where $e_{T_0}$ and $e_{T_1}$ are the errors of the children; $v_c$ is the vertex that splits the parent into its children; $z(v_c)$ is the field value assigned to $v_c$; and $z_T(v_c)$ is the field value that the parent assigns to the spatial location of $v_c$, equivalently, $z_T(v_c) = \frac{1}{2}(z(v_0) + z(v_1))$, where $v_0$ and $v_1$ are the vertices of the parent's split edge. This error is still a genuine bound on the difference between our representation and the representation of the original data set. However, it is looser than the bound computed directly from the data. The error computed from the children has the advantage that the error associated with a cell bounds not only the deviation from the original representation but also the deviation from the

representation at any intermediate resolution level. Consequently, this error is *nested* or *monotonic* in the sense that the error of a child is guaranteed not to be greater than the error of the parent. Once the errors of the leaf cells are computed, the nested bound for all cells higher in the tree can be computed in time proportional to $K$, where $K$ is the number of leaf cells in the tree. This can be accomplished by traversing the tree in a bottom up fashion.

The material interface error associated with a leaf node is the maximum value of the errors associated with each of the material interfaces in the node. For each material interface, the error is the maximum value of the errors associated with the vertices constituting the triangle mesh defining the interface and being inside the cell. The error of a vertex is the absolute value of the distance between the vertex and the interface approximation. The material interface error of **E** for a cell guarantees that no point in the original interface polygon mesh is further from its associated approximation that a distance of **E**. This error metric is an upper bound on the deviation of the original interfaces from our approximated interfaces. A cell that does not contain a material interface is considered to have an interface error of zero.

## 7 Results

We have tested our algorithm on a data set resulting from a simulation of a hypersonic impact between a projectile and a metal block. The simulation uses a logically rectilinear mesh of dimensions 32x32x52. For each cell, the average density and pressure values are available, as well as the per-material densities and volume fractions. The physical dimensions in x, y, and z directions are [0,12] [0,12] and [-16,4.8].

There are three materials in the simulation: the projectile, the block, and empty space. The interface between the projectile and the block consists of 38 polygons, the interface between the projectile and empty space consists of 118 polygons and the interface between empty space and the block consists of 17574 polygons. Figure 4 shows the original interface meshes determined from the volume fraction information. The green mesh is the interface between the metal block and empty space; the blue mesh is the interface between the projectile and empty space; and the red mesh is the interface between the projectile and the block. The meshes shown in Figure 4 were generate from a 17x17x28 data set.

Figures 5 shows a cross section view of the mesh created by a cutting plane through the tetrahedral mesh. The black lines are the original interface polygons intersected by the plane, and the magenta lines are our approximation to the interface. The interface approximation error is 0.15. An error of 0.15 means that all of the vertices in the original material interface meshes are no more that a physical distance of 0.15 from their associated interface approximation. This is equivalent to an error of (0.5 - 1.5)% when considered against the physical dimensions. A total of 3174 tetrahedra were required to approximate the interface to an error 0.15. The overall mesh contained a total of 5390 tetrahedra. A total of 11990 tetrahedra were required to approximate the interface to an error of 0.15 and the density field to an error of 3. The maximum field approximation error in the cells containing material interfaces is 2.84 and the average field error for these cells is 0.007. These error measurements indicate that separate field representations for the materials on either side of a discontinuity can accurately reconstruct the field.

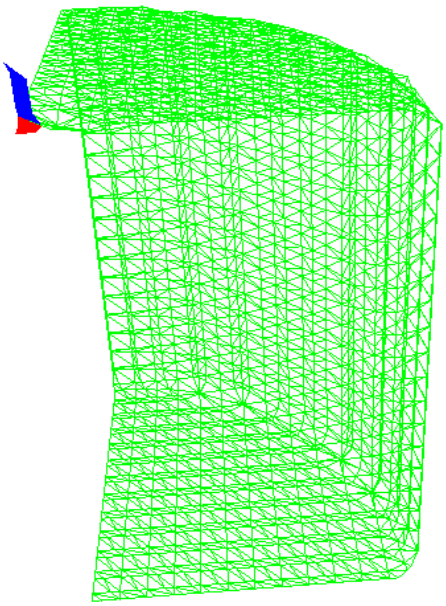

Figure 4: Original triangular meshes representing material interfaces.

Figures 6 and 7 compare the density fields generated using linear interpolation of the cell average density values and explicit field representations on either side of the discontinuity. Figure 7 shows that using explicit field representations in the presence of discontinuities can improve the quality of the field approximation. This can be seen in the flat horizontal and vertical sections of the block where the cells approximate a region that contains the block and empty space. In these cells, the use of explicit representations of the discontinuities leads to a exact representation of the density field. The corresponding field representations using linear interpolation, shown in Figure 6, do a very poor job of capturing the discontinuities. Furthermore, Figure 7 captures more of the dynamics in the area where the projectile is entering the block (upper left corner). The linear interpolation of the density values in the region where the projectile is impacting the block smooths out the density field, and does not capture the distinct interface between the block and the projectile. Figure 8 shows the density field from Figure 7 with our approximation to the interface and without the cell outlines.

## 8 Conclusions and Future Work

We have presented a simplification method for scientific data sets that explicitly represents material interfaces in mesh cells. Our algorithm constructs an approximation that can be used in place of the original data set for visualization purposes. Explicitly representing the material and implicit field discontinuities allows us to use multiple field representations to better approximate the field within each cell. The use of the tetrahedral subdivision allows us to generalize our algorithm to a wide variety of data sets and to support interactive level-of-detail exploration and view-dependent simplification. Future work will extend our error calculations to support complex native data element types such as tetrahedra and curvilinear hexahedra. Our current ghost value com-

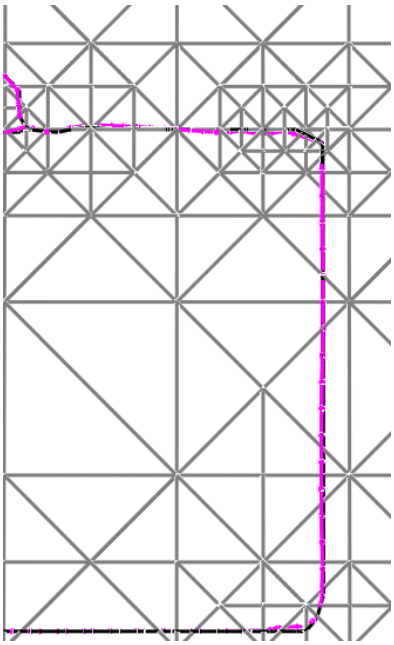

Figure 5: Cross section of the tetrahedral mesh showing the original interfaces and interface approximations.

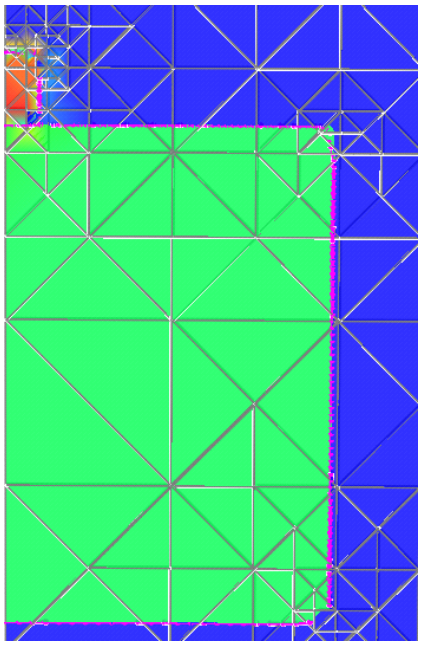

Figure 7: Density field using explicit interface representations and separate field representations (interface error = 0.15).

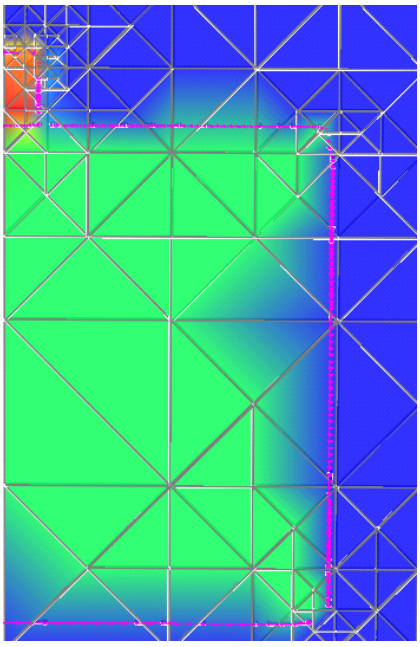

Figure 6: Density field using linearly interpolated cell average density values field (interface error = 0.15).

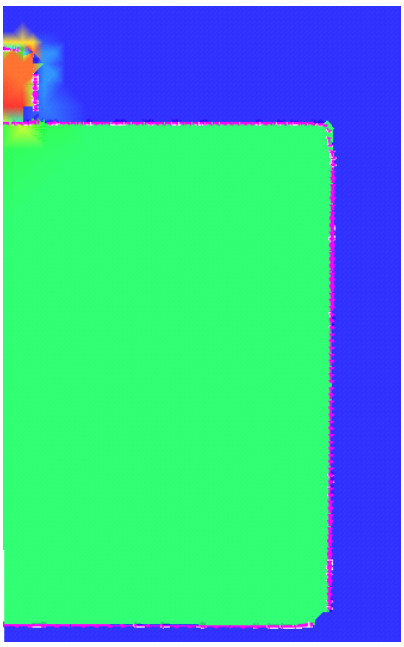

Figure 8: Figure 7 without the cell outlines.

putation assumes that the field is constant on the other side of the interface. Higher-order extrapolation methods should be investigated for ghost value computation to determine if a superior field approximation can be obtained. Similarly, material interfaces are defined by approximations based on linear functions. The tradeoff between cell count and higher-order approximation methods should be investigated to determine if a better approximation can be obtained without a great increase in computational complexity. Finally, we plan to apply our algorithm to more complex unstructured data sets.

## 9 Acknowledgments

This work was supported by the National Science Foundation under contracts ACI 9624034 and ACI 9983641 (CAREER Awards), through the Large Scientific and Software Data Set Visualization (LSSDSV) program under contract ACI 9982251, and through the National Partnership for Advanced Computational Infrastructure (NPACI); the Office of Naval Research under contract N00014-97-1-0222; the Army Research Office under contract ARO 36598-MA-RIP; the NASA Ames Research Center through an NRA award under contract NAG2-1216; the Lawrence Livermore National Laboratory under ASCI ASAP Level-2 Memorandum Agreement B347878 and under Memorandum Agreement B503159; and the North Atlantic Treaty Organization (NATO) under contract CRG.971628 awarded to the University of California, Davis. We also acknowledge the support of ALSTOM Schilling Robotics, Chevron, General Atomics, Silicon Graphics, and ST Microelectronics, Inc. We thank the members of the Visualization Group at the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis and the members of the Advanced Computing Laboratory at Los Alamos National Laboratory.

## References

[1] Y. Zhou, B. Chen, and A. E. Kaufman, "Multiresolution tetrahedral framework for visualizing regular volume data," in *IEEE Visualization '97* (R. Yagel and H. Hagen, eds.), pp. 135–142, IEEE Computer Society Press, Nov. 1997.

[2] H. Hoppe, "Progressive meshes," in *SIGGRAPH 96 Conference Proceedings* (H. Rushmeier, ed.), ACM SIGGRAPH, Addison Wesley, 1996.

[3] M. A. Duchaineau, M. Wolinsky, D. E. Sigeti, M. C. Miller, C. Aldrich, and M. B. Mineev-Weinstein, "ROAMing terrain: Real-time optimally adapting meshes," in *IEEE Visualization '97*, 1997.

[4] B. Heckel, A. E. Uva, B. Hamann, and K. Joy, "Surface reconstruction using adaptive clustering methods," in *IEEE Transactions on Visualization and Computer Graphics*, 2000.

[5] B. Heckel, A. E. Uva, and B. Hamann, "Clustering-based generation of hierarchical surface models," in *Proceedings IEEE Visualization '98 – Late Breaking Hot Topics*, 1998.

[6] P. Cignoni, E. Puppo, and R. Scopigno, "Multiresolution Representation and Visualization of Volume Data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, pp. 352–369, Oct. 1997.

[7] I. J. Trotts, B. Hamann, K. I. Joy, and D. F. Wiley, "Simplification of tetrahedral meshes," in *Proceedings IEEE Visualization '98*, pp. 287–296, IEEE Computer Society Press, Oct. 18–23 1998.

[8] A. Liu and B. Joe, "Quality local refinement of tetrahedral meshes based on bisection," *SIAM Journal on Scientific Computing*, vol. 16, pp. 1269–1291, Nov. 1995.

[9] O. G. Staadt and M. H. Gross, "Progressive tetrahedralizations," in *Proceedings of Visualization 98* (D. Ebert, H. Hagen, and H. Rushmeier, eds.), pp. 397–402, IEEE Computer Society Press, Oct. 1998.

[10] R. Klein, G. Liebich, and W. Straßer, "Mesh reduction with error control," in *Proceedings of IEEE Visualization '96*, pp. 311–318, IEEE Computer Society Press, Oct. 1996.

[11] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *SIGGRAPH 97 Conference Proceedings* (T. Whitted, ed.), Annual Conference Series, pp. 209–216, ACM SIGGRAPH, Addison Wesley, Aug. 1997.

[12] K. Bonnell, "On material boundary surfaces," m.s. thesis, Department of Computer Science, University of California, Davis, June 2000.

[13] K. Bonnell, M. A. Duchaineau, D. R. Schikore, B. Hamann, and K. I. Joy, "Constructing material interfaces from data sets with volume-fraction information," in *Proceedings of IEEE Visualization '00*, p. (submitted), 2000.